(19) Canadian Intellectual Property Office
An Agency of Industry Canada

Office de la Propri,t, Intellectuelle du Canada
Un organisme d'Industrie Canada

(11) CA 2 299 850 (13) A1

(40) 01.09.2001
(43) 01.09.2001

(12)

(21) 2 299 850

(22) 01.03.2000

(51) Int. Cl.7: G06F 17/60, G11B 23/00, G06F 11/36

(71)
MITEL, INC.,
400-205 Van Buren Street, HERNDON, XX (US).

(72)
CAVA, PHIL (US).

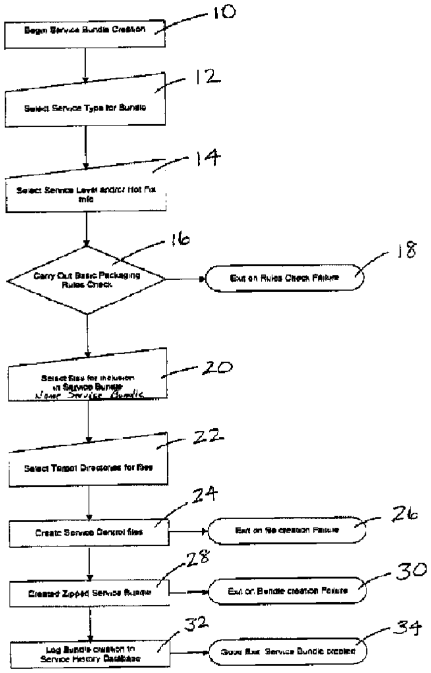
(74)
SIM & MCBURNEY

(54) SYSTEME ET METHODE DE GESTION DE MAINTENANCE LOGICIELLE

(54) SYSTEM AND METHOD FOR THE MANAGEMENT OF COMPUTER SOFTWARE MAINTENANCE

(57)

The present invention is a system and method for managing at least one computer software update. The system and method comprise the steps of creating a service bundle comprising at least one software update; retrieving and storing the service bundle on a target system; and executing and hacking actions of the at least one software update on the target system.


10
Begin Service Bundle Creation
12
Select Service Type for Bundle
14
Select Service Level and/or Hot Fix info
16
Carry Out Basic Packaging Rules Check
18
Exit on Rules Check Failure
20
Select files for inclusion in Service Bundle
Name Service Bundle
22
Select Target Directories for files
24
Create Service Control files
26
Exit on file creation Failure
28
Created Zipped Service Bundle
30
Exit on Bundle creation Failure
32
Log Bundle creation in Service History Database
34
Good Exit Service Bundle created

Office de la Propriété Intellectuelle du Canada
Un organisme d'Industrie Canada

Canadian Intellectual Property Office
An agency of Industry Canada

CA 2299850 A1 2001/09/01

(21) 2 299 850

(12) DEMANDE DE BREVET CANADIEN
CANADIAN PATENT APPLICATION

(13) A1

(22) **Date de dépôt/Filing Date:** 2000/03/01

(41) **Mise à la disp. pub./Open to Public Insp.:** 2001/09/01

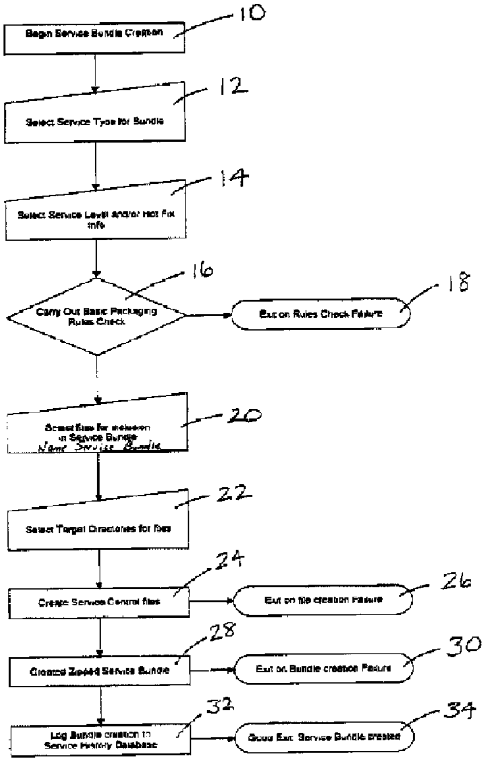
(51) **Cl.Int.^{7}/Int.Cl.^{7}** G06F 17/60, G06F 11/36, G11B 23/00

(71) **Demandeur/Applicant:**
MITEL, INC., US

(72) **Inventeur/Inventor:**
CAVA, PHIL, US

(74) **Agent:** SIM & MCBURNEY

(54) Titre : SYSTEME ET METHODE DE GESTION DE MAINTENANCE LOGICIELLE
(54) Title: SYSTEM AND METHOD FOR THE MANAGEMENT OF COMPUTER SOFTWARE MAINTENANCE


Begin Service Bundle Creation
10
Select Service Type for Bundle
12
Select Service Level and/or Hot Fix info
14
Carry Out Basic Packaging Rules Check
16
Exit on Rules Check Failure
18
Select files for inclusion in Service Bundle
Name Service Bundle
20
Select Target Directories for files
22
Create Service Control files
24
Exit on file creation Failure
26
Created Zipped Service Bundle
28
Exit on Bundle creation Failure
30
Log Bundle creation to Service History Database
32
Good Exit Service Bundle created
34

(57) Abrégé/Abstract:
The present invention is a system and method for managing at least one computer software update. The system and method comprise the steps of creating a service bundle comprising at least one software update; retrieving and storing the service bundle on a target system; and executing and hacking actions of the at least one software update on the target system.

SYSTEM AND METHOD FOR THE MANAGEMENT OF COMPUTER SOFTWARE MAINTENANCE

Abstract

The present invention is a system and method for managing at least one computer software update. The system and method comprise the steps of creating a service bundle comprising at least one software update; retrieving and storing the service bundle on a target system; and executing and tracking actions of the at least one software update on the target system.

SYSTEM AND METHOD FOR THE MANAGEMENT OF COMPUTER SOFTWARE MAINTENANCE

Field of the Invention

The present invention relates in general to computer software maintenance and more specifically to a system and method for managing updates to be applied to computer software products.

Background of the Invention

Software vendors are constantly updating their computer software products to maintain market and customer acceptance. The software updates may add new features to the computer software product, remove unnecessary files and/or correct existing problems, or bugs. In some cases, software updates are released on a regular basis by vendors. As a result, keeping track of the software updates can be difficult. Also, many vendors release different versions of the same computer software product. Different software updates may therefore be required for different releases.

Presently, software vendors typically use one software program to create, test and package software updates and a separate software program to release and install the software updates. Using separate programs to test and package software updates and to release and install software updates, results in software updates being released to consumers without validating the appropriateness of the software updates for individual consumers.

It is therefore an object of the present invention to provide a novel system and method for managing updates to be applied to computer software products.

Summary of the Invention

In order to overcome the problems in the prior art, the system and method of the present invention provide a single software module to manage the packaging and tracking of software updates for multiple versions of a software application on a service system and to manage the installation, removal and tracking of software updates for multiple versions of a software application on a target system to maintain system integrity.

By including tools to track the execution of software updates on a target system, the present invention is able to monitor the updates and provide assistance in case an error occurs during execution. If an error occurs, execution of the update may be restarted from the error point instead of from the beginning to provide automatic recovery from a failed update install.

Furthermore, the system and method of the present invention provide tools to verify that the software update is applicable to the target system in order to maintain system integrity.

According to an aspect of the invention, there is provided a method for managing updates to be applied to a computer software product comprising the steps of:

using a system tool on a vendor computer site to create and package an update for a computer software product of the vendor; and

using the same tool to apply the update to a corresponding computer software product on a customer's computer site.

According to another aspect of the invention there is provided a system for managing updates to be applied to a computer software product comprising:

a customer computer site; and

a service bundle for executing said at least one computer software update on said customer computer site;

wherein said service bundle comprises:

said at least one software update; and

tracking means for monitoring application of the update to verify integrity of the computer software product.

In yet another aspect of the invention there is provided a computer readable medium including computer program code for managing updates to be applied to a computer software product, said computer readable medium comprising:

computer program code for executing an update on a customer computer site;

computer program code for monitoring application of the update to verify integrity of the computer software product;

computer program code for halting application of the update if a system integrity error is detected; and

computer program code for fixing the system integrity error.

In a further aspect of the present invention, there is provided a method of managing at least one computer software update comprising the steps of:

creating a service bundle comprising said at least one software update;

distributing said service bundle to a customer computer site;

executing said at least one software update on said customer computer site and tracking actions of said execution to verify and maintain system integrity.

Brief Description of the Detailed Drawings

A preferred embodiment of the present invention will now be described more fully with reference to the accompanying drawings, in which like numerals denote like parts throughout, and in which:

Figure 1 is a schematic diagram of a service bundle;

Figure 2 is a flowchart showing a service bundle creation process;

Figure 3 is a flowchart showing a service bundle installation process;

Figure 4 is a flowchart showing a service bundle removal process;

Figure 5 is a flowchart showing a self-service file process;

Figure 6 is a flowchart showing a basic service rules check process;

Figure 7 is a flowchart showing a service file relationship checking process;

Figure 8 is a flowchart showing a loadtime failure recovery process;

Figure 9 is a flowchart showing a recover on pre service command failure process;

Figure 10 is a flowchart showing a recover on post service command failure process;

Figure 11 is a flowchart showing a recover on failed file copy process; and

Figure 12 is a flowchart showing a recover on failed database transaction process.

Detailed Description of the Preferred Embodiment

As mentioned above, software vendors typically release software updates to customers on a continuing basis to enhance computer software products. Generally, the software updates correct existing and/or foreseeable bugs, enhance applications and/or add new features. To manage computer software product updates in this environment, the present invention provides a system and method for packaging, tracking, installing and removing computer software updates for a computer software product having multiple versions. A preferred embodiment of the present invention will now be described.

The present invention provides a service system tool to allow computer product updates or "service items" to be created, tracked and packaged on a vendor's computer site and to allow computer software products on customer computer sites to be tracked and updated with appropriate service items. The service system tool is operable in a bundler mode and in a tool mode. Specifically, the service system tool is conditioned to the bundler mode on the vendor's computer site when it is desired to create, manage and package service items for installation on customers' computer sites. The service system tool is conditioned to the tool mode when it is desired to install and track the service items on customers' computer sites. The service system tool also includes a command line interface to record service item installations in a service history database. In the tool mode, the service history database includes computer product version information and applied service item information. In the bundler mode, the service history database tracks service items available to versions of the computer software products.

In the bundler mode, the service system tool creates, manages and packages service items into service bundles. Turning to Figure 1, a schematic representation of a service bundle is shown and is generally indicated to by reference numeral 1. As can be seen, service bundle 1 includes a service type information section 3, which identifies the type of software update, or updates in the service item that is stored in the service bundle. The service bundle 1 also includes a service level/hot fix information section 4. Section 4 includes information concerning the service status of the service bundle i.e. Beta version, Alpha version or released version. Section 4 also includes information relating to smallest service item within the service bundle.

The service bundle 1 also includes a files/updates section 5 storing service item files making up the service item. A service control files section 6 stores control files associated with the service items files. The service control files are used to control the operation of the various service types of the service bundle. The name of each service control file is based on the service level and a description of the service item file. In the present embodiment, the service control files can include service map files, service command files, service history database entry files and service fix list files.

Service map files list the service item files included in the service bundle 1 and the target directory for each service item file. During installation of the software update, outdated files, from previous updates, are archived and replaced by the update files. The service map files store the location of the archived files in case the archived files have to be restored due to an error during the installation of the software update. This property is intended to maintain system integrity.

The service command files aid in the parsing of contents of service commands and for executing service commands. Examples of service commands include the installation and removal programs used in connection with the service bundle. The service history database entry files store pertinent information concerning updates in a database entry file. It also stores information in a bundler section of the service history database during service bundle creation and information in a tool section of the service history database during installation of the service bundle. The service history database entry file also maintains information relating to the installation/removal of service bundles.

Service fix list files determine the service item files contained in the service bundle and place the service item files in a service fix list during service bundle creation. The service fix list control file uses the service fix list during installation or removal of the service bundle.

The service bundle 1 also includes a name field 7 identifying the service bundle. The name of the service bundle 1 is created by combining the service level information and a description of the service item, to yield a file name that is unique for the service bundle. The file extension of the service bundle indicates whether the service bundle includes new service item to be installed or is to be re-install archived service files. In the preferred embodiment, a .CSB extension denotes a service bundle to install new service item files while a .CSA extension denotes a service bundle to re-install archival service item files.

The service bundle 1 also includes a target directory representing the location on the computer site where the software update is to be stored and executed.

Turning to Figure 2, a flow chart displaying the steps to create a service bundle using the system service tool is shown. It will be understood that during creation of the service bundle, the system service tool is conditioned to the bundler mode. Firstly, at step 10, creation of the service bundle is initiated. Afterwards, at step 12, the service type of the service bundle is selected. If the service item is to contain service item files relating to more than one software update, more than one service type may be selected. Service level and/or hot fix information is then selected for the service bundle at step 14 and a basic packaging rules check is executed at step 16 to verify that the service level of the service bundle matches the service level of the computer software product to which the service bundle is to be applied. If the service bundle fails the basic packaging rules check, service bundle creation is halted (step 18).

If the service bundle creation passes the basic packaging rules check, the appropriate service item files making up the service item are selected for inclusion in the service bundle and the service bundle is named (step 20). Target directories for the service item files are then selected (step 22). Afterwards, a set of service control files is created (step 24). If an error occurs during the creation of the service control files, service bundle creation is halted (step 26). Otherwise, the service bundle is compressed and stored in a zipped file format (step 28). If an error occurs during service bundle compression, service bundle creation is halted (step 30). Once the zipped service bundle is created, the zipped service bundle is logged and stored in the service history database (step 32) and the service bundle creation program is exited (step 34).

Once the service bundle has been created and stored in the service history database, the service bundle may be distributed to the target computer site for installation.

Figure 6 better illustrates the steps performed during the basic packaging rules check. During the basic packaging rules check, the service bundle is first tested for failed service recovery (step 159). The test is performed to determine service bundle is able to support recovery from errors during the installation or removal process. If the service bundle passes this test, the basic packaging rules check is exited (step 160). If the service bundle fails the test, the service level of the service bundle is determined (step 162). If a bad service bundle is detected, the basic packaging rules check is exited (step 164) and the service bundle creation is stopped (step 168). Otherwise, the current service level of the service bundle is determined from the service history database (step 166). If a database error is detected, the basic packaging rules check is exited (step 168) and the service bundle creation is halted (step 18). If no database error is detected, the service bundle level is compared with the service level required by the target computer site (step 170).

This comparison is used to verify that the service item stored in the service bundle is appropriate for the target computer site. If no match is found, the basic rules packaging check is exited (step 172) and the service bundle creation is halted (step 18). Otherwise, the service bundle is considered to have basic packaging rules check and the rules check is exited (step 173).

Turning to Figure 3, a flowchart outlining the process for installing the service bundle on a customer computer site using the service system tool is shown. It will be understood that during this process the service bundle is conditioned to the tool mode. Initially, the target computer site accesses the vendor computer site through a dial-up or

Internet connection and intitializes the installation process (Step 35). At this point, the service bundle to be applied to the target computer site is selected (step 36).

A basic service rules check is then performed (step 38) to verify that the service bundle is appropriate for the target computer site. If the service bundle fails the basic service rules check, installation of the service bundle is halted (step 40). Otherwise, the service control files are extracted from the service bundle (step 42). If the service control files are not correctly extracted, installation of the service bundle is halted (step 44). If no errors arise during extraction of the service cortrol files, the service control files are checked for self-service, or self-executable files (step 46). In most cases, icons are depressed in order to execute a program but, in some cases, files are self-executing and therefore, no icons need to be depressed. If a self-service file is detected, the file is immediately executed, by a self-service module, at step 48 as will be described.

If there is no self-service file present, the service control files are parsed (step 50) to determine the service bundle level along with its contents. If a failure occurs while parsing the service control files, installation of the service bundle is halted (step 52). Otherwise, a service files relationship rules check is performed (step 53). If the service files relationship rules check fails, service bundle installation is stopped (step 54). Otherwise, outdated service item files, from previous service bundles, are replaced with new service item files (step 56). At this point, the service bundle changes from a .CSB extension to a .CSA extension. The outdated service item files that have been replaced are archived. This allows the computer software product to be restored. Information concerning the location of the archived files and also their original location is stored in the service map file. If the file archiving process fails, service bundle installation is halted (step 58)

Following the file archiving process, pre-apply service commands are executed (step 60). If a failure occurs, a pre-apply service command failure recovery program is executed at step 62 to correct the failure. The pre-apply service command failure recover program will be described below. Otherwise, each of the service item update files is copied to its target directory (step 64). If a failure occurs, a file copy failure recovery program is executed at step 66 to correct the file copy failure. The file copy failure recovery program is described below. If the service item files are correctly copied, post-apply service commands are executed (step 68). Failure to execute the post-apply service commands results in the execution of a post-apply service command failure recovery program at step 70 to correct the service command error. The post-apply service command recovery program is described below.

Upon successful execution of the post-apply service commands, service bundle installation is checked to determine if it is a result of a failed service recovery (step 72). If it is a result of a failed service recovery, installation of the service bundle is complete (step 74). Otherwise, the service history database is updated to indicate that the service bundle was successfully applied to the target computer site on the date/time, at step 76. If an error occurs during the service history database update, a database transaction failure recovery program is executed at step 78. If the service history database is successfully updated, installation of the service bundle is complete (step 80). As will be appreciated, each step of the installation process of the service bundle is tracked to ensure integrity. If an error occurs during installation, recovery programs are executed as will be described further herein.

Figure 4 is a flow chart detailing the steps for removing a service bundle from a target computer site. Once, the removal process has been initiated (step 81), the service bundle to be removed is selected (step 82). A basic service rules check is then performed on the selected service bundle (step 84). If the service bundle fails the basic service rules check, service bundle removal is halted (step 86). Otherwise, the service control files are extracted from the service bundle (step 88). Service bundle removal is also terminated if an error occurs during extraction of the service control files (step 90). If no error occurs, the service bundle is checked for self-service files (step 92). If a self-service file exists, the service bundle removal program enters a self-service operation module (step 94). Otherwise, the service control files are parsed (step 96). If an error occurs during the service control file parsing, service bundle removal is stopped (step 98). If no error occurs, a service files relationship rules check is performed (step 100). If the service files relationship rules check fails, service bundle removal is halted (step 102).

If the service files relationship rules check passes, pre-remove service commands are executed (step 104). If a failure occurs during command execution, a pre-remove service command failure recovery program is executed at step 106. If no failure occurs, each of the service files is copied to its target directory (step 108). If a failure occurs, a file copy failure recovery program is executed at step 110. Otherwise, post-remove service commands are executed (step 112). If the post-remove service commands cannot be executed, a post-remove service command failure recovery program is executed at step 114. If the post-remove service commands are successfully executed, the service bundle execution is monitored to determine if it is a result of a failed service recovery occurred (step 116). If it is a result of a failed service recovery, the service bundle removal is considered to be completed and the process is exited (step 118). Otherwise, the service history database is updated to indicate that the service bundle was successfully removed, at step 120. If an error occurs during the service history database update, a database transaction failure recovery program is executed at step 122. If the service history database is successfully updated, the service bundle removal process is completed (step 124).

Figure 5 is a flow chart showing the steps performed to deal with self-service files. When a self-service file is detected, the self-service module is initiated (step 125). The self-service module then attempts to obtain service mutex from the target computer site (step 126). This causes the actions of the service bundle to be mutually exclusive and run independently with respect to other computer applications. If the self-service module is unable to obtain service mutex, execution of the self-service file is halted (step 128). Otherwise, the target computer site registry is checked to monitor the progress of the request for service mutex. (step 130). If the target computer site returns a message of nothing to do or uncorrectable failure, execution of the self-service file is halted (step 132). If service mutex is received, the self-service file is extracted from the service bundle (step 134). An extraction failure results in the termination of self-service file execution (step 136). Otherwise, the service control files are parsed (step 138).

If an error occurs during parsing of the service control files, self-service file execution is halted (step 139). If no error occurs, a service file relationships rules check is performed (step 140). Failure to pass the service file relationships rules check, results in the termination of self-service file execution (step 142). Upon passing the service file relationships rules check, the outdated service files from previous service bundles that are replaced are replaced by the self-service files are archived (step 144). If the file archive process fails, execution of the self-service file is halted (step 146). Otherwise, at step 148, the self-service file is copied to its target directory (step 148). If a failure occurs during copying, the file copy failure recovery program is executed (step 150). If no copying errors occur, the service system tool is launched to complete service operation by installing the service item file (step 152). If the service system tool is unable to launch, execution is halted (step 154). Once the service operation has been completed the service mutex is returned to the target computer site (step 156) and the self-service module is exited (step 157).

In order to monitor and verify integrity, the system service tool utilizes two separate methodologies. One methodology is based on rules based service operations and the other is based on automatic failed service recovery. The implementation of these methodologies causes an integrity check to be performed at almost every step during the creation, installation and/or removal of a service bundle.

The rules based service operations perform checks during all service operations to ensure that target computer site integrity is not compromised as a result of the installation or removal of a service bundle. These operations are specific to service operation types. The basic packaging rules are implemented so that no service bundle is created, which duplicates a service bundle stored in the service history database. This is achieved by using a unique service bundle naming procedure. System integrity is also maintained since the contents of the service bundle cannot be changed after the service bundle has been created and logged in the service history database.

The steps performed during the service files relationships rules check are shown in Figure 7. This check is performed to verify that application of the service bundle does not cause a regression of a previous service item and/or to verify that installation of the service bundle does not cause improper removal of a pre-requisite service item. This check also monitors the actions of files during operation of the installation of the service bundle.

Once initiated (step 174), a failed service recovery test is performed (step 174). If this test is passed, the service bundle returns to the function from which it exited (step 176). If the test is not passed, the service bundle begins a loop to search through the service bundle files for the service map control file (step 178). Version information is then read from the target computer site and the service item files (step 180). If a file read error occurs, the rules check is exited (step 182). Otherwise, the archived service item files, stored on the target system site, are compared to the service item files, stored in the service bundle (step 184). If a required software service item file is missing from the service bundle, a check is performed to see if the service item file is being installed or removed (step 186). If a match is found, the service rules check is exited due to the presence of a regression (step 188). Otherwise, the target prerequisites are compared with the service item files in the service bundle (step 190).

After comparing the target computer site prerequisites, if a prerequisite is missing, a check is performed to see if the missing prerequisite is being installed or removed (step 191). If a match is found, the service rules check is exited due to the presence of the missing prerequisite (step 192). Otherwise, the prerequisites of the service item are compared with the prerequisites in the archived service item files (step 194). This step also occurs after the comparison between the target prerequisites and the service item files when no missing prerequisite is located. Prerequisites are used to verify that the installation of the service bundle is appropriate for the target computer site. For instance, if a target site is running version 5 of an application and the service bundle is providing an update to version 4, then installation of the service bundle is not appropriate.

If there are any unmet prerequisites, a check is performed to see if the missing prerequisite is the one being applied or removed (step 196). If the result is negative, the self rules check is exited (step 198). Otherwise, more checks are performed to see if any other service item files need to be checked (step 200). If more files need to be checked, the module returns to step 178 to search through the remaining service item files listed in the service map control file. Otherwise the self rules check is passed and a message is returned to the function from which it exited (step 202).

The service system tool also verifies the appropriateness of a service bundle, as well as, correctness of execution based on the automatic recovery from failed service methodology. Examples are shown with respect to Figures 8 to 12.

Figure 8 shows a loadtime recovery process to monitor the service bundle installation and removal processes for errors. If an error occurs, the loadtime recovery process attempts to solve the problem and allow, upon fixing of the problem, the service bundle installation or removal to continue from the point where the error occurred instead of requiring the process to restart. Once the recovery process is initiated (step 203), a test is performed to check for the presence of any other recovery processes (step 204). If another recovery process is being executed, the present loadtime recovery process is halted (step 206). Otherwise, a test for service in progress is performed on the target computer site (step 208). If the loadtime recovery process does not locate any errors, the recovery process is halted since no recovery is necessary (step 210). Otherwise, a test is performed to find out what type of recovery is necessary (step 212).

If no recovery appears necessary, the loadtime recovery process is stopped (step 214). Otherwise, a failed service operation (step 215) and a failed service step (step 216) are invoked. The failed service operation (step 215) determines which process was being executed when the error occurred and the failed service step (step 216) determines at which step the error occurred. A check is then performed to see if the failure is recoverable (step 217). If recovery is not possible, the loadtime recovery process is halted and installation or removal of the service bundle is restarted (step 218). Otherwise, the loadtime recovery process locates where the error occurred (step 219) and executes routines necessary to fix the error. This includes, but is not limited to the failed pre-service command recovery program (step 220), the failed post service command recovery program (step 222), the failed service file copy recovery program (step 224) and/or the failed database transaction recovery program (step 226).. Afterwards, the load time recovery process verifies that the recovery was successful (step 228).

As described above, during installation or removal of a service bundle, various failure recovery processes are executed based on event failure. One such process is the pre-service command failure recovery as shown in the flowchart of Figure 9.

Upon initiation of the command failure recovery process (step 233), the service bundle is checked to see if the function from which it exited was the installation process (step 234). If the function is not the installation process, the post apply service command is executed at step 236. Otherwise, the post remove service command is executed at step 238. After execution of either service command, a check is performed to verify that the service command was successfully executed (step 240). A message is then returned to the function from which it exited indicating whether execution of the service command execution was successful (step 242) or a failure (step 244). If the service command execution is successful, the function from which it exited continues from the point where the error occurred.

The post-service command failure process is shown in Figure 10. Once the post-service command failure process is initiated (step 245), the service bundle is checked to see if the function from which it exited was the installation of the service bundle (step 246). If the function from which it exited is not installation, the service bundle is re-instated (step 248). Otherwise, the execution of the service bundle returns to the start of the remove service (step 250). After execution of either service command, a check is executed to verify that the service command was successfully executed (step 252). A message is then returned to the function from which is exited indicating whether the service command execution was successful (step 254) or a failure (step 256). If the service command execution is successful, the function from which it exited continues from the point where the error occurred.

The service file copy failure recovery process is shown in Figure 11. Once this recovery process is initiated (step 257), the service bundle is checked to see if the function from which it exited was installation of the service bundle (step 258). If the function from which it exited is not installation, the service bundle installation returns to the apply service (step 260). Otherwise, the execution of the service bundle returns to the start of the remove service (step 262). After execution of either service command, a check is executed to verify that the service operation was successfully executed (step 264). A message is then returned to the function from which it exited indicating whether the service command execution was successful (step 266) or a failure (step 268). If the service command execution is successful, the function from which it exited continues from the point where the error occurred.

The database transaction failure recovery process is shown in Figure 12. After initiation of the database recovery process (step 269), the service bundle is checked to see if the recovery mode was inline (step 270). An inline recovery process is a method to write progress and diagnostic information during a service operation in a form from which a failing operation can be determined. If the function from which it exited is not inline, a partial record is removed from the database (step 272). Otherwise, the entire transaction is roll-backed and restarted (step 274). After execution of either database transaction, a check is executed to verify that database integrity is restored (step 276). A message is then returned to the function from which it exited indicating whether the database integrity has been restored (step 278) or not restored (step 280).

It will be appreciated that, although only one embodiment of the invention has been described and illustrated in detail, various changes and modification may be made. All such changes and modifications may be made without departing from the spirit and scope of the invention as defined by the claims appended herein.

What is Claimed Is:

1. A method for managing updates to be applied to a computer software product comprising the steps of:

using a system tool on a vendor computer site to create and package an update for a computer software product of the vendor; and

using the same tool to apply the update to a corresponding computer software product on a customer's computer site.

2. The method of Claim 1 wherein the system tool operates in a bundler mode during creation and packaging of the update and operates in a tool mode during application of the update.

3. The method of Claim 2 wherein the system tool packages updates in a service bundle, said service bundle including at least one service item file and a target directory for said at least one service item file.

4. The method of Claim 3 further comprising the step of monitoring application of the update to verify integrity of the corresponding computer software product.

5. The method of Claim 4 further comprising the step of archiving files of said computer software product that are removed during application of the update to enable the computer software product to be restored.

6. The method of Claim 1 further comprising the step of storing said update in a database after the step of creating and packaging the update.

7. The method of Claim 3 wherein the step of using a system tool on a vendor computer site to create and package an update for a computer software product of the vendor comprises the steps of:

selecting a service type for said service bundle;

selecting a service level and/or hot fix information for said service bundle;

selecting an update for said service bundle;

naming said service bundle; and

creating service control files for said service bundle.

8. The method of Claim 4 wherein the step of monitoring the application of the update to verify integrity of the computer software product includes searching for self-service files, checking the customer computer site for previous updates, verifying database updates and passing of rules checks.

9. A system for managing updates to be applied to a computer software product comprising:

a customer computer site; and

a service bundle for executing said at least one computer software update on said customer computer site;

wherein said service bundle comprises:

said at least one software update; and

tracking means for monitoring application of the update to verify integrity of the computer software product.

10. The system of Claim 9 further comprising a database for storing the service bundle.

11. The system of Claim 10 wherein the service bundle further comprises at least one service item file and a target directory for said at least one service item file.

12. The system of Claim 11 wherein said at least one service item file is a service map file, a service command file, a service history database entry file or a service fix list file.

13. A computer readable medium including computer program code for managing updates to be applied to a computer software product, said computer readable medium comprising:

computer program code for executing an update on a customer computer site;

computer program code for monitoring application of the update to verify integrity of the computer software product;

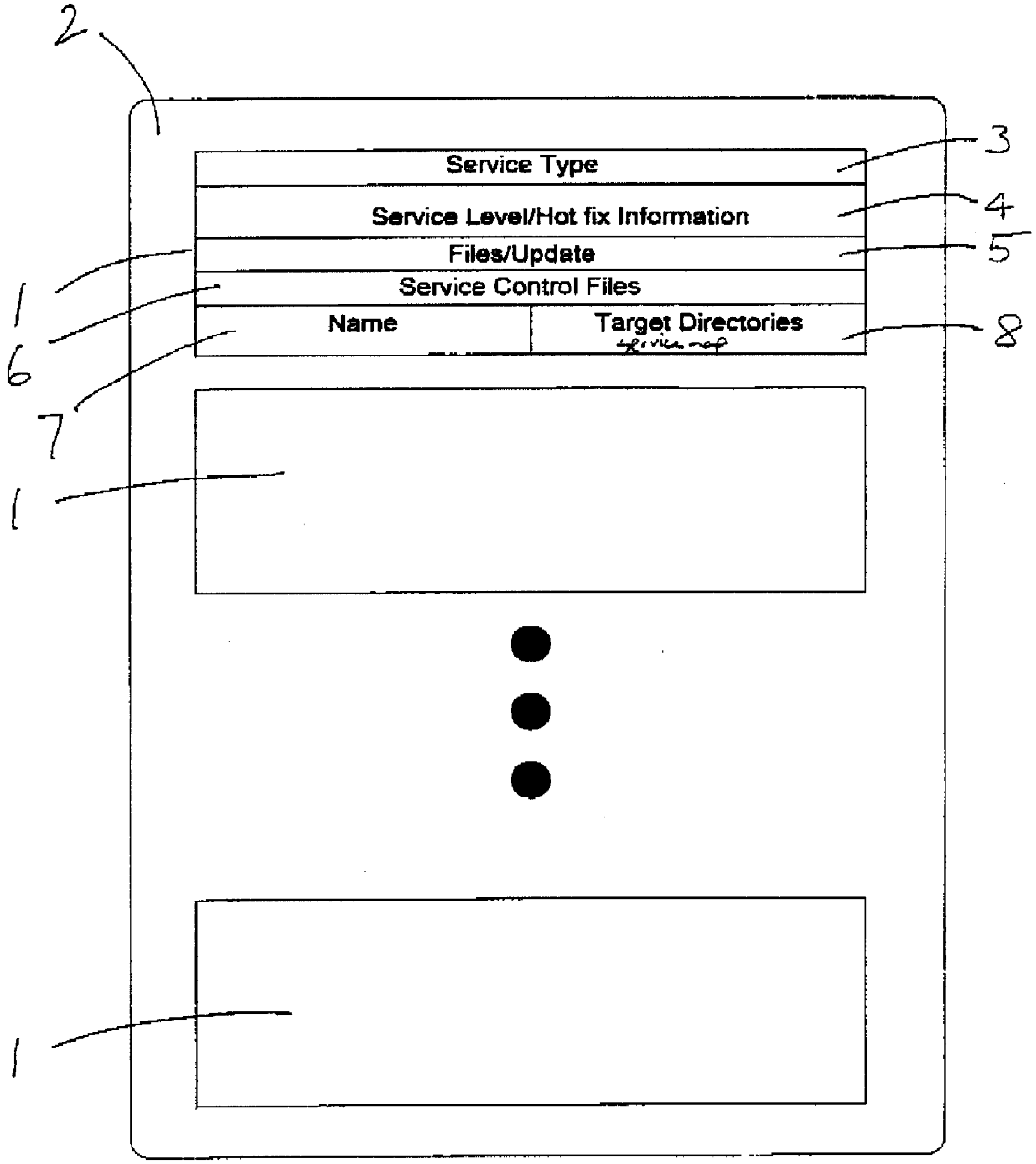
computer program code for halting application of the update if a system integrity error is detected; and computer program code for fixing the system integrity error.

14. The computer readable medium of Claim 13 wherein monitoring of the application of the update includes at least one of searching for errors, searching for self-service files, checking said customer computer site for previous updates, verifying database updates and passing of rules checks.

15. The computer readable medium of Claim 14 further comprising:
computer program code for controlling execution of the update.

16. A method of managing at least one computer software update comprising the steps of:
creating a service bundle comprising said at least one software update;
distributing said service bundle to a customer computer site;
executing said at least one software update on said customer computer site and tracking actions of said execution to verify and maintain system integrity.

FiG. 1


2
3
4
5
1
6
7
8
Service Type
Service Level/Hot fix Information
Files/Update
Service Control Files
Name
Target Directories
service [illegible]
1
1

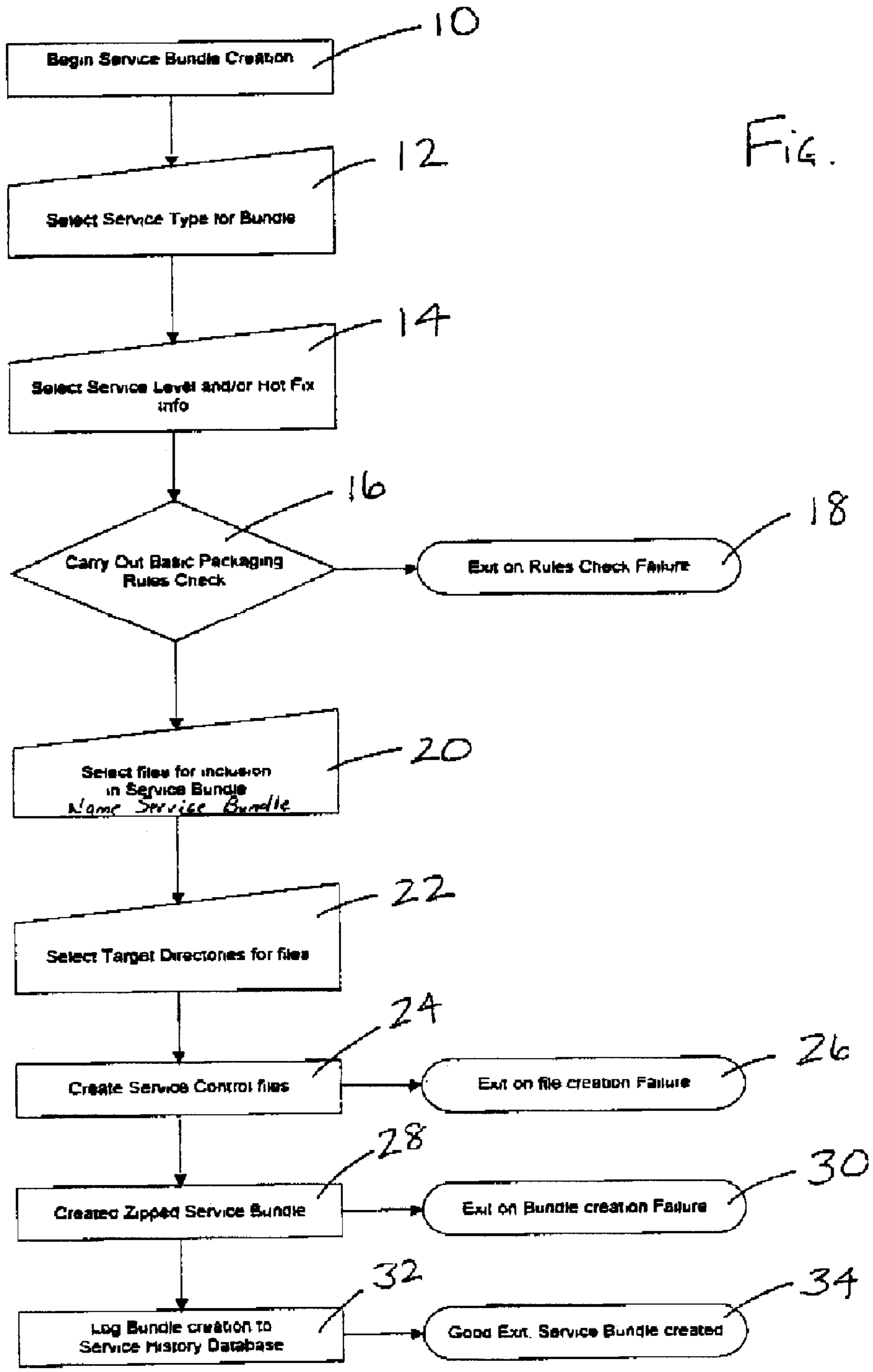

10
Begin Service Bundle Creation
12
Select Service Type for Bundle
14
Select Service Level and/or Hot Fix info
16
Carry Out Basic Packaging Rules Check
18
Exit on Rules Check Failure
20
Select files for inclusion in Service Bundle
Name Service Bundle
22
Select Target Directories for files
24
Create Service Control files
26
Exit on file creation Failure
28
Created Zipped Service Bundle
30
Exit on Bundle creation Failure
32
Log Bundle creation to Service History Database
34
Good Exit. Service Bundle created

Fig. 2

Fig. 3

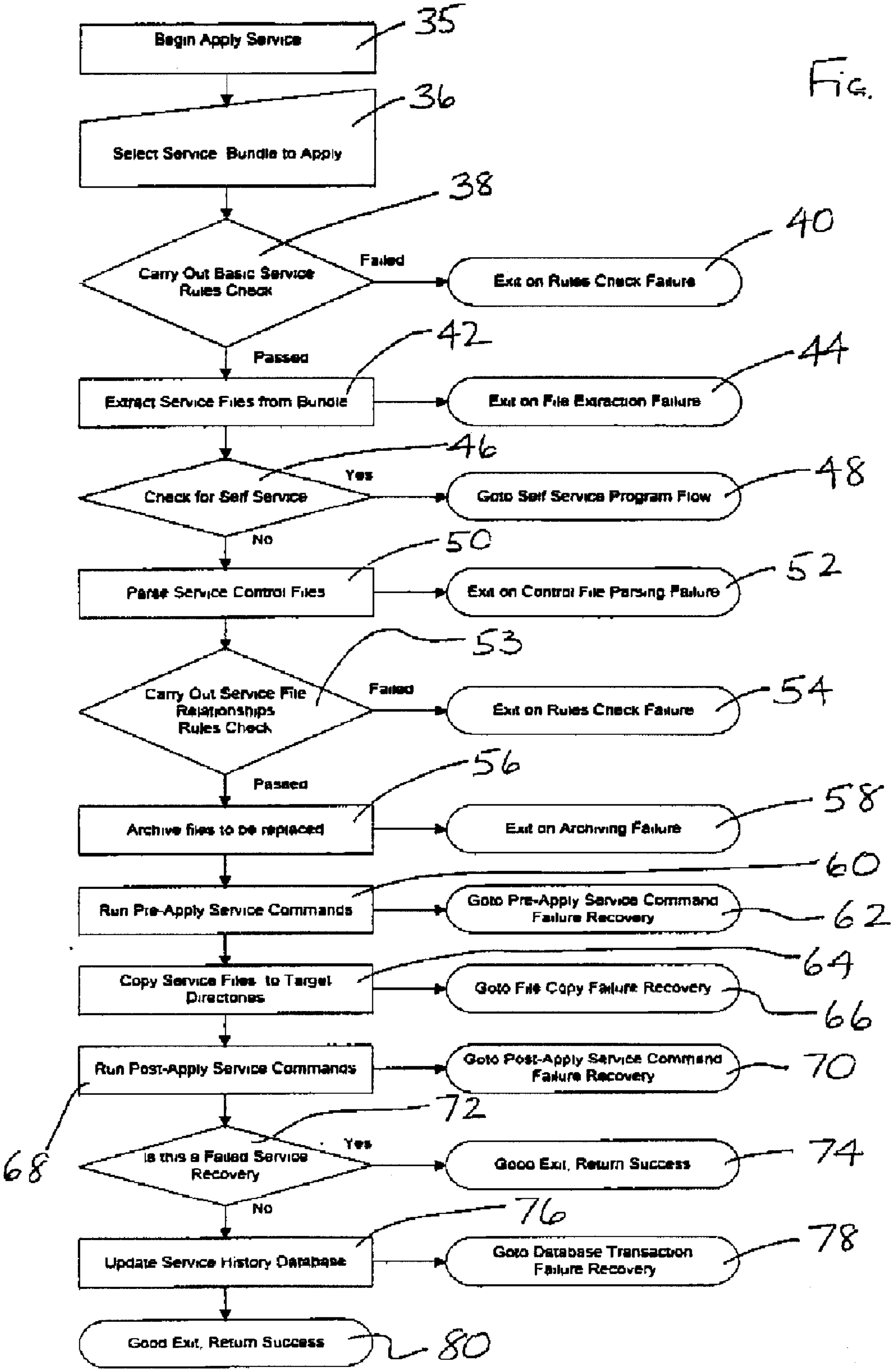
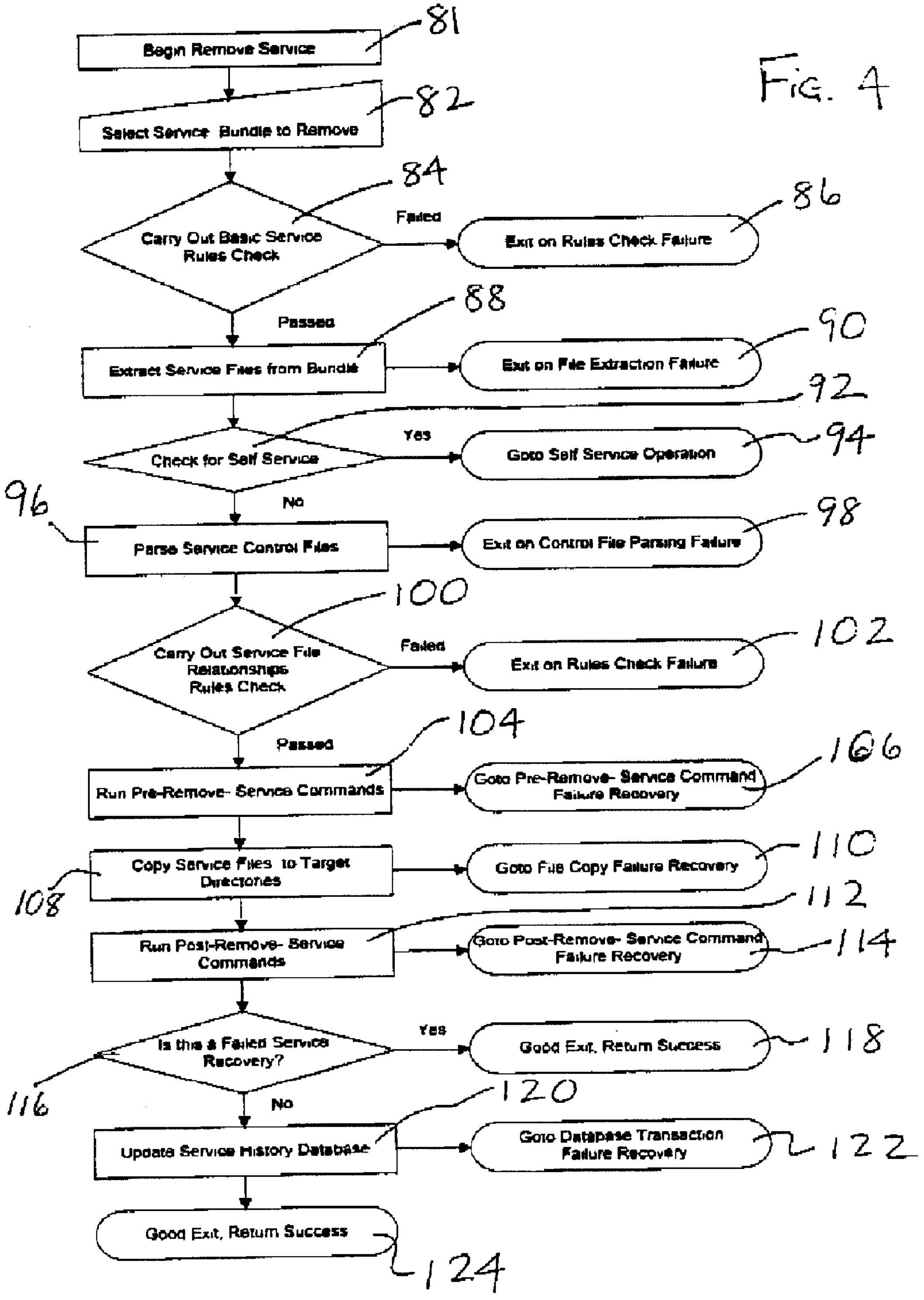

Begin Apply Service
35
Select Service Bundle to Apply
36
Carry Out Basic Service Rules Check
38
Failed
Exit on Rules Check Failure
40
Passed
Extract Service Files from Bundle
42
Exit on File Extraction Failure
44
Check for Self Service
46
Yes
Goto Self Service Program Flow
48
No
Parse Service Control Files
50
Exit on Control File Parsing Failure
52
Carry Out Service File Relationships Rules Check
53
Failed
Exit on Rules Check Failure
54
Passed
Archive files to be replaced
56
Exit on Archiving Failure
58
Run Pre-Apply Service Commands
60
Goto Pre-Apply Service Command Failure Recovery
62
Copy Service Files to Target Directories
64
Goto File Copy Failure Recovery
66
Run Post-Apply Service Commands
68
Goto Post-Apply Service Command Failure Recovery
70
Is this a Failed Service Recovery
72
Yes
Good Exit, Return Success
74
No
Update Service History Database
76
Goto Database Transaction Failure Recovery
78
Good Exit, Return Success
80

Fig. 4


81
Begin Remove Service
82
Select Service Bundle to Remove
84
Carry Out Basic Service Rules Check
Failed
86
Exit on Rules Check Failure
Passed
88
Extract Service Files from Bundle
90
Exit on File Extraction Failure
92
Check for Self Service
Yes
94
Goto Self Service Operation
No
96
Parse Service Control Files
98
Exit on Control File Parsing Failure
100
Carry Out Service File Relationships Rules Check
Failed
102
Exit on Rules Check Failure
Passed
104
Run Pre-Remove- Service Commands
166
Goto Pre-Remove- Service Command Failure Recovery
108
Copy Service Files to Target Directories
110
Goto File Copy Failure Recovery
112
Run Post-Remove- Service Commands
114
Goto Post-Remove- Service Command Failure Recovery
116
Is this a Failed Service Recovery?
Yes
118
Good Exit, Return Success
No
120
Update Service History Database
122
Goto Database Transaction Failure Recovery
Good Exit, Return Success
124

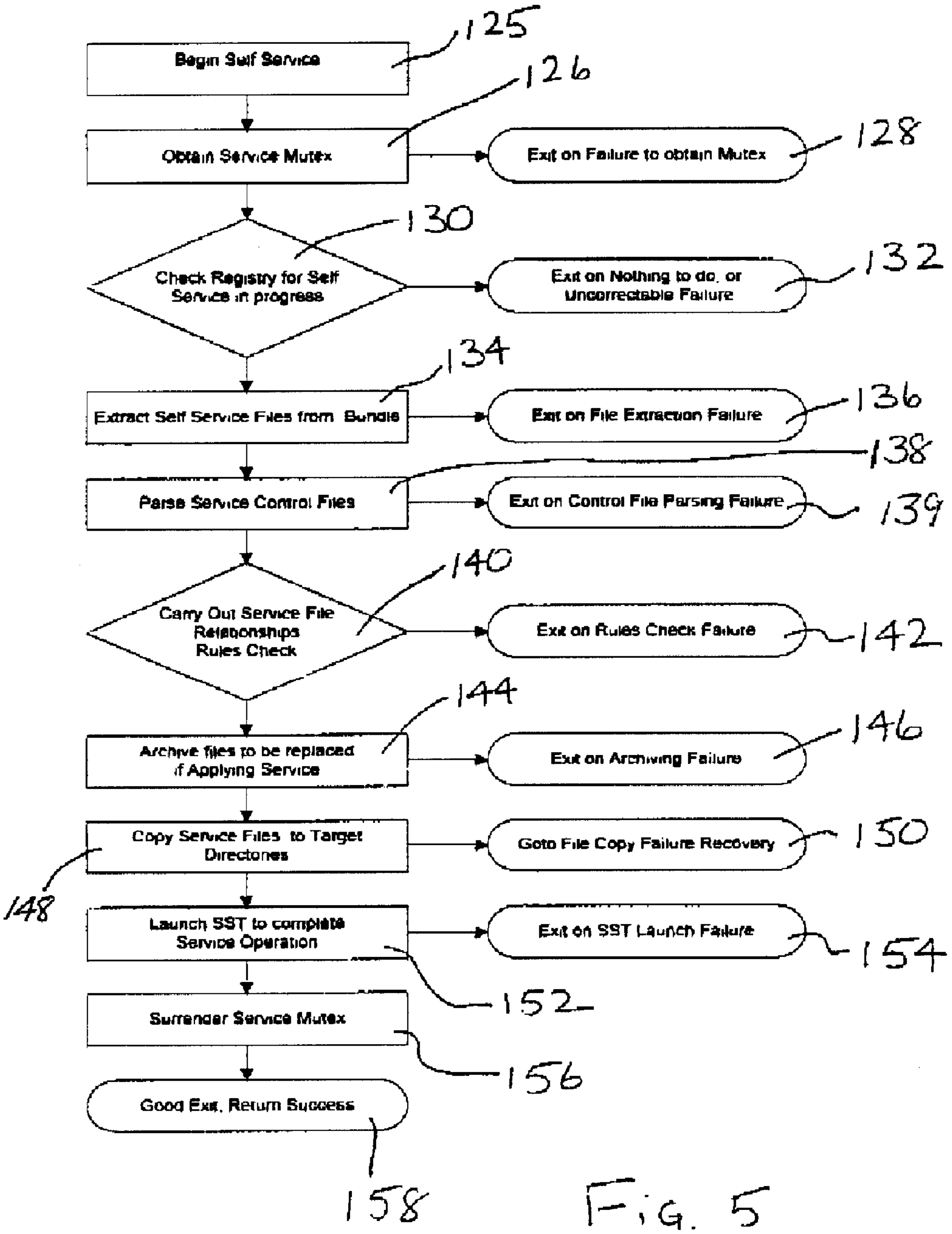

Begin Self Service
125
Obtain Service Mutex
126
Exit on Failure to obtain Mutex
128
Check Registry for Self Service in progress
130
Exit on Nothing to do, or Uncorrectable Failure
132
Extract Self Service Files from Bundle
134
Exit on File Extraction Failure
136
Parse Service Control Files
138
Exit on Control File Parsing Failure
139
Carry Out Service File Relationships Rules Check
140
Exit on Rules Check Failure
142
Archive files to be replaced if Applying Service
144
Exit on Archiving Failure
146
Copy Service Files to Target Directories
148
Goto File Copy Failure Recovery
150
Launch SST to complete Service Operation
152
Exit on SST Launch Failure
154
Surrender Service Mutex
156
Good Exit, Return Success
158

Fig. 5

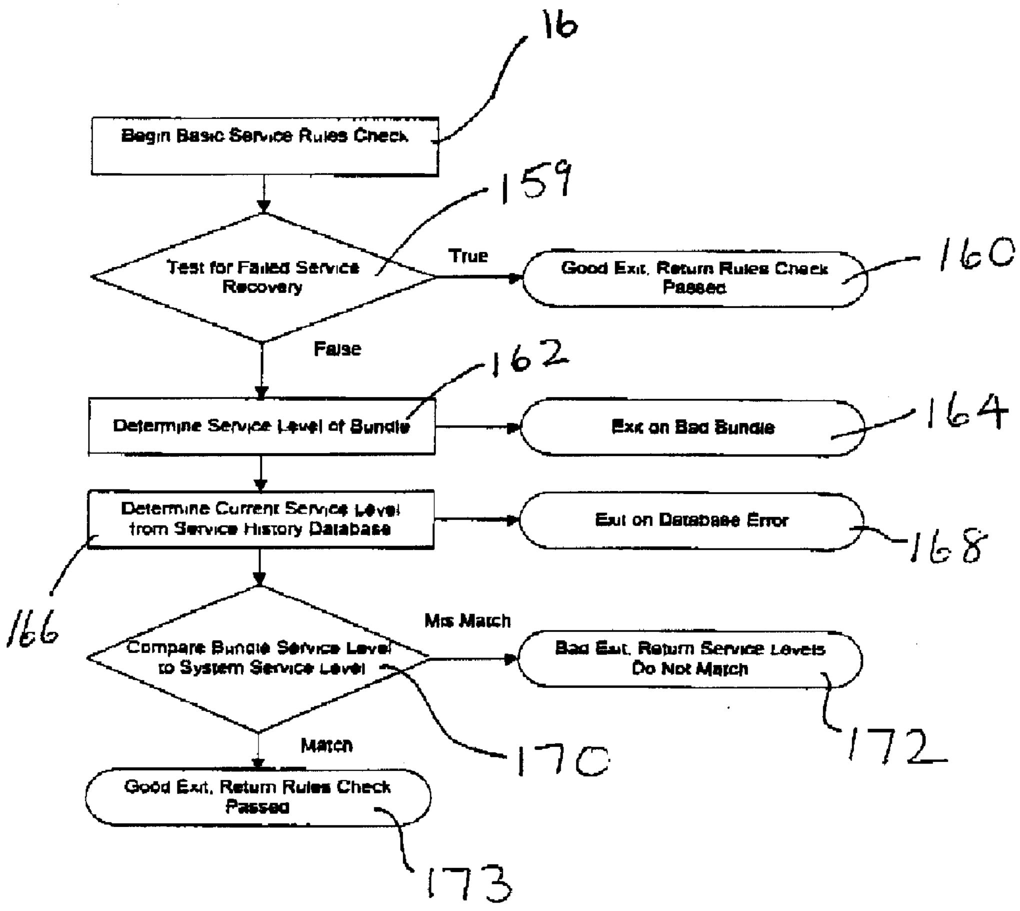

16
Begin Basic Service Rules Check
159
Test for Failed Service Recovery
True
Good Exit, Return Rules Check Passed
160
False
162
Determine Service Level of Bundle
Exit on Bad Bundle
164
Determine Current Service Level from Service History Database
Exit on Database Error
168
166
Mis Match
Compare Bundle Service Level to System Service Level
Bad Exit, Return Service Levels Do Not Match
170
172
Match
Good Exit, Return Rules Check Passed
173

FIG. 6

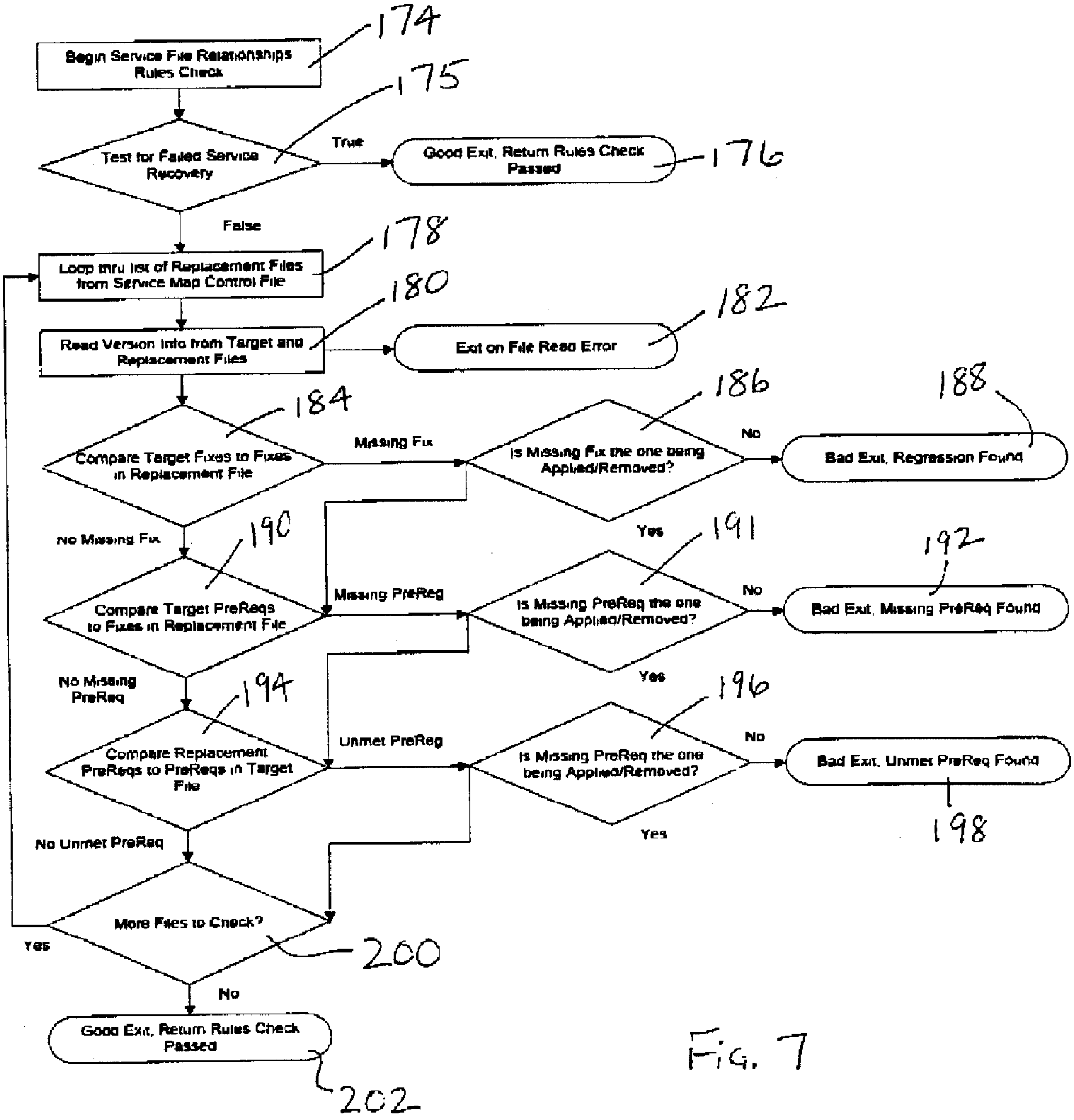

174
Begin Service File Relationships Rules Check
175
Test for Failed Service Recovery
True
Good Exit, Return Rules Check Passed
176
False
178
Loop thru list of Replacement Files from Service Map Control File
180
Read Version Info from Target and Replacement Files
182
Exit on File Read Error
184
Compare Target Fixes to Fixes in Replacement File
Missing Fix
186
Is Missing Fix the one being Applied/Removed?
No
188
Bad Exit, Regression Found
Yes
No Missing Fix
190
Compare Target PreReqs to Fixes in Replacement File
Missing PreReq
191
Is Missing PreReq the one being Applied/Removed?
No
192
Bad Exit, Missing PreReq Found
Yes
No Missing PreReq
194
Compare Replacement PreReqs to PreReqs in Target File
Unmet PreReq
196
Is Missing PreReq the one being Applied/Removed?
No
Bad Exit, Unmet PreReq Found
198
Yes
No Unmet PreReq
More Files to Check?
200
Yes
No
Good Exit, Return Rules Check Passed
202

Fig. 7

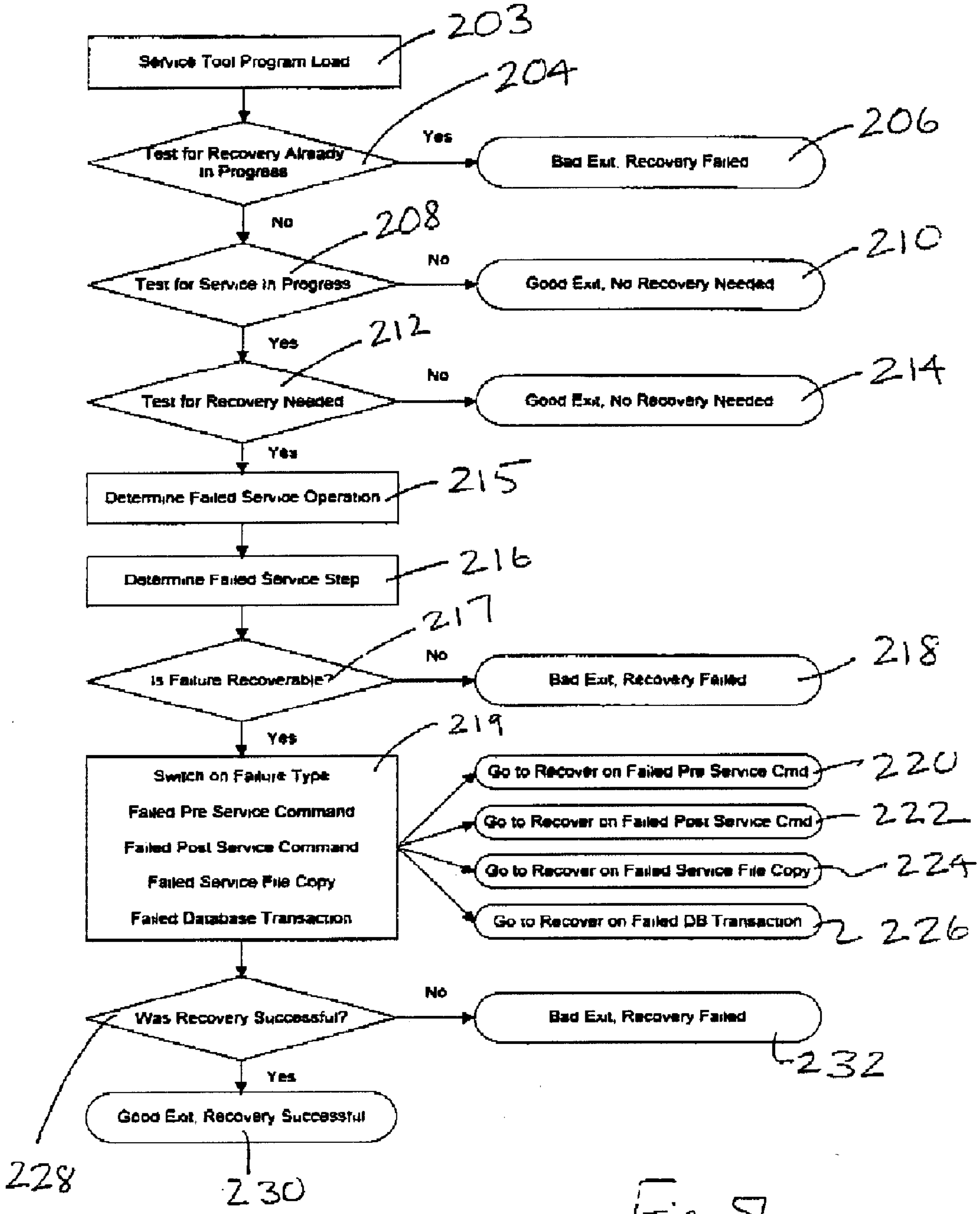

Service Tool Program Load
203
Test for Recovery Already In Progress
204
Yes
Bad Exit, Recovery Failed
206
No
Test for Service In Progress
208
No
Good Exit, No Recovery Needed
210
Yes
Test for Recovery Needed
212
No
Good Exit, No Recovery Needed
214
Yes
Determine Failed Service Operation
215
Determine Failed Service Step
216
Is Failure Recoverable?
217
No
Bad Exit, Recovery Failed
218
Yes
Switch on Failure Type
Failed Pre Service Command
Failed Post Service Command
Failed Service File Copy
Failed Database Transaction
219
Go to Recover on Failed Pre Service Cmd
220
Go to Recover on Failed Post Service Cmd
222
Go to Recover on Failed Service File Copy
224
Go to Recover on Failed DB Transaction
226
Was Recovery Successful?
No
Bad Exit, Recovery Failed
232
Yes
Good Exit, Recovery Successful
228
230

Fig. 8

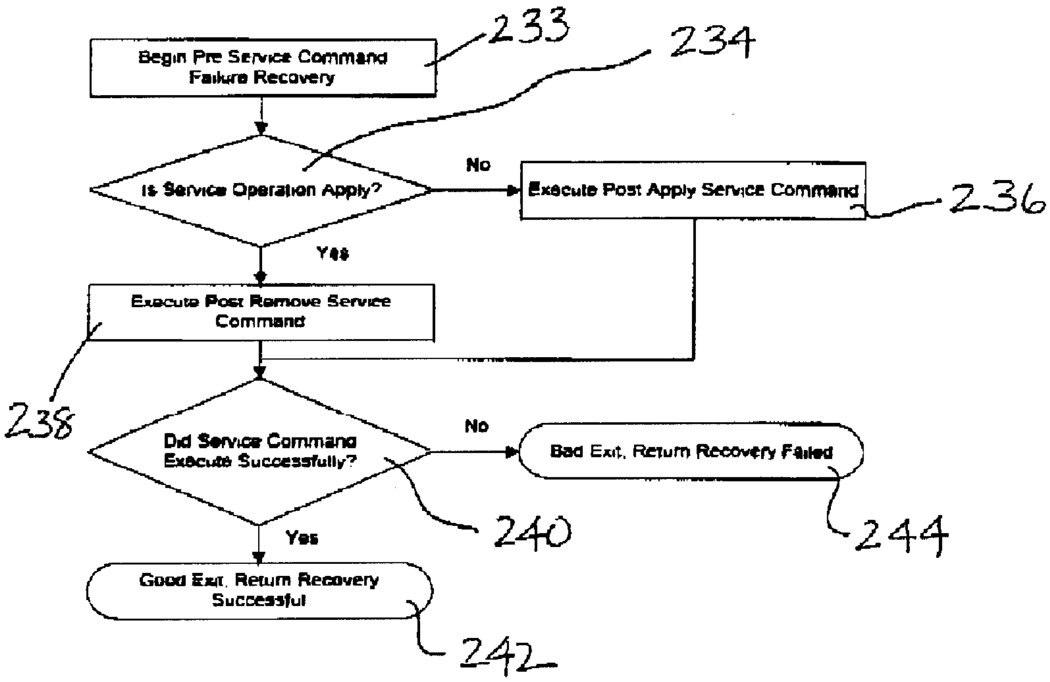

233
234
Begin Pre Service Command Failure Recovery
Is Service Operation Apply?
No
Execute Post Apply Service Command
236
Yes
Execute Post Remove Service Command
238
Did Service Command Execute Successfully?
No
Bad Exit, Return Recovery Failed
244
Yes
240
Good Exit, Return Recovery Successful
242

Fig. 9

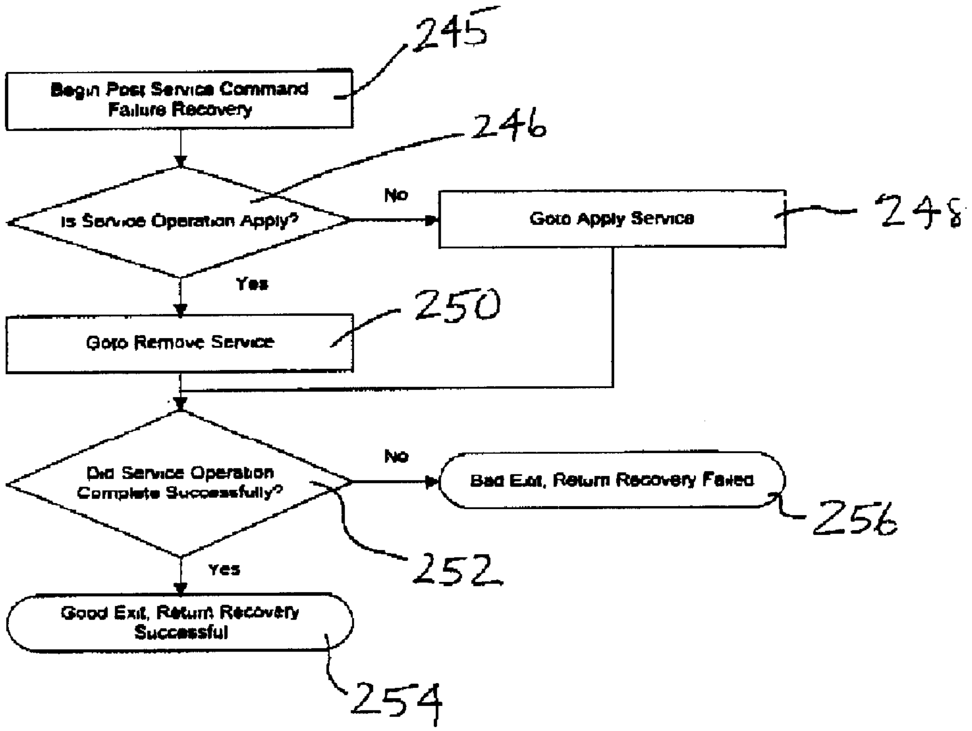

245
Begin Post Service Command Failure Recovery
246
Is Service Operation Apply?
No
Goto Apply Service
248
Yes
Goto Remove Service
250
Did Service Operation Complete Successfully?
No
Bad Exit, Return Recovery Failed
256
252
Yes
Good Exit, Return Recovery Successful
254

Fig. 10

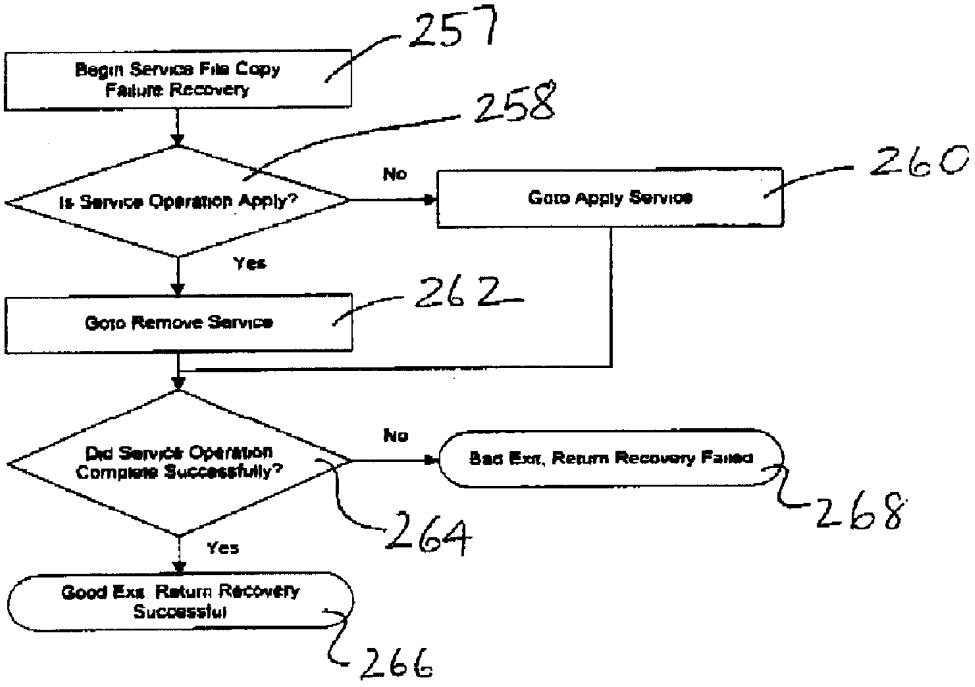

Begin Service File Copy Failure Recovery
257
258
Is Service Operation Apply?
No
Goto Apply Service
260
Yes
Goto Remove Service
262
Did Service Operation Complete Successfully?
No
Bad Exit, Return Recovery Failed
268
264
Yes
Good Exit Return Recovery Successful
266

Fig. 11

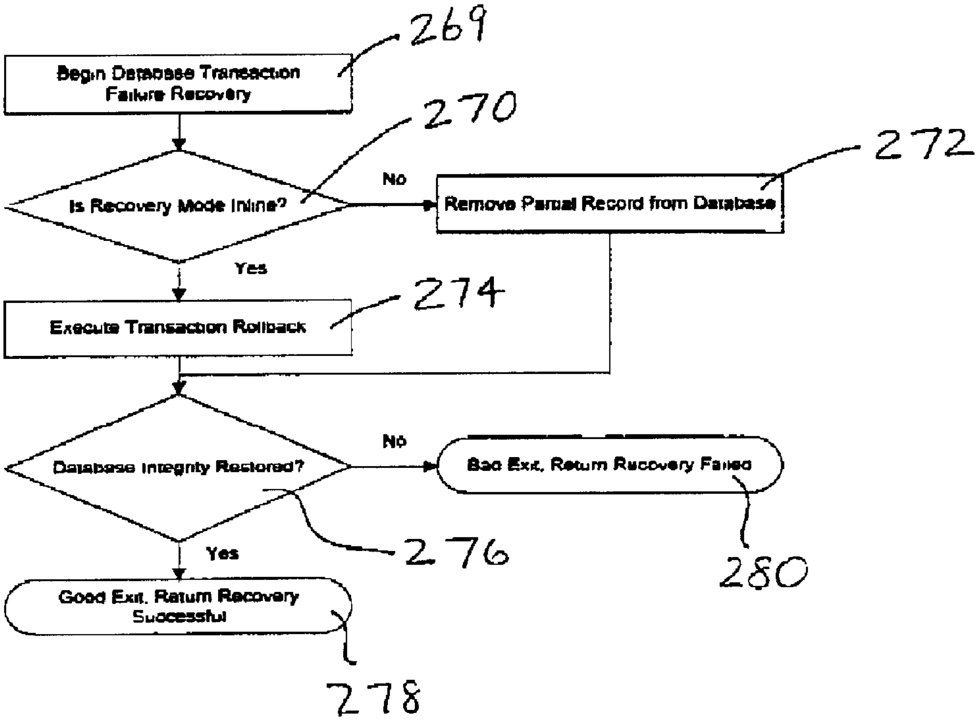

Begin Database Transaction Failure Recovery
269
Is Recovery Mode Inline?
270
No
Remove Partial Record from Database
272
Yes
Execute Transaction Rollback
274
Database Integrity Restored?
No
Bad Exit, Return Recovery Failed
280
276
Yes
Good Exit, Return Recovery Successful
278

Fig. 12